TS1000 Doc.
2068 Romm set to "64" Directly
Replaces older 2364 TS1000 Romm.
-JJO

BUILD A 2764 BASED "CARTRIDGE" BOARD AND PROGRAMMER

One of the biggest "headaches" for Timex/Sinclair enthusiasts is the painfully slow cassette LOADing performed by the ZX-81/TS1000 computer. This, and the fact that one little "glitch" can make us have to do it all over again, prompted me to design this 2764 based "software on eprom" project.

At the time of designing my 2716 programmer and reader, as previously published in SQ no.1, the 2716 was the cheapest per bit eprom on the market. This has now drastically changed. The current price of a 2716 (450ns) is about $3.50ea, while the 2764 (300ns) sells for only $5.75 ea, and is still dropping in price! Four times the memory capacity for less than twice the price. At this kind of price, storing BASIC programs on 2764 eproms is approaching the economics of using cassette tapes for program storage. And, when you consider that once the program is SAVEd on eprom, we can instantly "download" it, rather than having to wait paitiently on our cassettes, and that every "download" will LOAD in instantly and PERFECTLY, as compared to the LOADing problems associated with cassette tapes, it makes you want to throw your whole cassette library away!!!

I am not, however, suggesting anything like this. What I am suggesting, though, is storing our most commonly used programs on eprom cartridges, and using the cassettes for our least used programs. This is the goal of our project this time.

First, let's cover the specs of our project. The 2764 programmer is very similar to the 2716 programmer shown in SQ no.2. It is mapped from 2000-3FFFH (8192-16393D) and contains it's own timer and latches, so it is compatible with any dynamic memory you may be using. (IE. The Z-80 WAIT NOT input line is not used) It programs in exactly the same way as the 2716 programmer, by POKing one of it's location the desired data and using a PAUSE 4 statement after each POKE. It may also be programmed, via a simple USR command, with the current BASIC program in the computers' memory. Also, like the 2716 programmer, to keep it simple, (remember KISS?) it does not have verify circuitry.

A 21VDC regulated supply is required for Vpp, unlike the 25VDC required by the 2716. I have included the schematic for the recommended Vpp supply, along with a single sided PC board layout for the same. (Which, if you aren't into etching, can be ordered from the author) If, however, you own about any adjustable regulated supply, capable of covering the voltage range from 5VDC to 21VDC, it may also be used.

The 2764 reader (cartridge) board will hold two Intel pinout 2764s, and is capable of mapping either one anywhere in memory on 8K blocks. In this project, however, we will map our eproms from C000-FFFFH. There is also a "board enable" switch onboard to deselect our eproms. Thus, we may have more than one cartridge in our expansion board at once, with only the board desired enabled.

Our cartridge is capable of holding more than one program per board. Each program does not have a "name", per se, but is selected by it's program number. The first program's number is 0, the second programs' number is 1, the thirds' is 2, etc.. The desired programs' number should be POKEd to memory location 16417 before calling the download routine.

The routines' default program will be program 0, because on power up 16417=0.

The BASIC programs should be stored on eprom without any separating bytes, one immediately following the other. The download program will calculate where they are and how long they are. All programs are stored on eprom exactly as they are stored on tape, except they are nameless. (IE. From 4009H (VERSN) to (4014/4015H)-1 (ELINE-1))

The machine language eprom programming program will take care of "listable" programs if the heading instructions given with it are followed. This program should be stored permanently in memory somewhere in the 2000-3FFFH memory block on another 2764 or 2716. Both this program and the downloader program are relocateable, so you can have them in memory anyplace you like as far as they are concerned. The downloader, of course, cannot be held in a BASIC REM statement because the downloading itself would wipe it out. (Self destructing software!) Really, the best place for both these programs is permanently in the 2000-3FFFH memory block on eprom.

For "unlistable" programs, the only way to get them on eprom is to load them in from tape with a program that has data SAVE/LOAD features, such as HOT Z. Instructions for doing just this with HOT Z are given in another section of this article.

Now, on to the hardware! In order to use the 2764 programmer, the Timex/Sinclair ROM must be fully decoded, to open up this new space. If you have built my 64K RAM from SQ no.3, or the 2716 programmer/readers from SQ no.s 1+2, then this has already been done. If you own any other 64K rampak, then this would have already been done, too. If you have only 16K of ram, and have NOT bought or built anything to fully decode the rom, then you can use the 2764 reader board to do this for you. There is sufficient logic onboard to decode the ROM, but you must remember that every time you use the programmer you must also have this board in your expansion board, too. To use the 2764 reader to decode the ROM, simply jumper the "0" output (pin 15) to the ROM CS NOT trace (Sinclair no. 23B) at the boards edge connector traces. Solder the wire carefully, only at the very top of the trace. Now, whenever this board is in your expansion board, the ROM will be fully decoded. It does not matter if the board's enable switch is off or on.

To map the board from C000-FFFFH, this space must obviously be open for use. If you own my 64K memory, then all that is required is to turn off the 48/64 switch on the board. If you use a Timex/Sinclair 16K ram, then you must fully decode it. See SQ no.1 for details on this. If you own any other brand of 64K memory, then you must figure out some way to disable the top 16K. The circuit (enclosed), although untested, will probably do the job for you.

As you can see, owners of my 64K ram really hbve it easy. It's all been done for you!

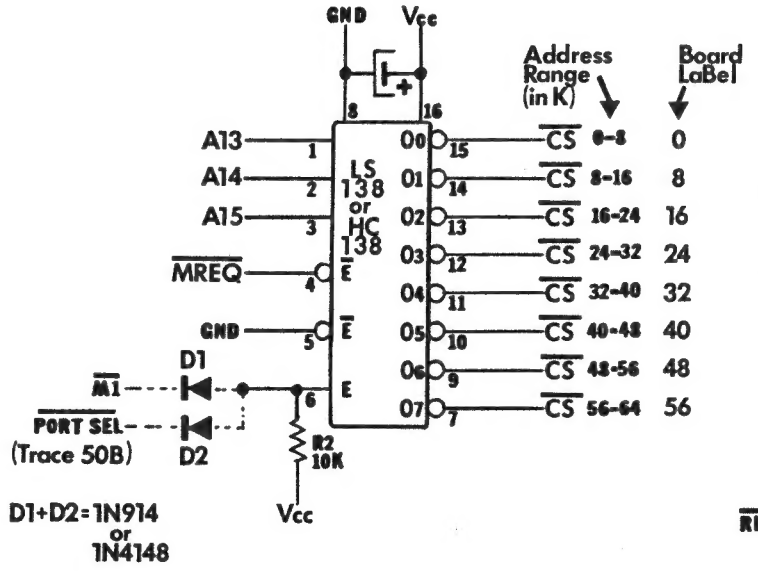
Now, let's actually build our boards. You may etch your own boards from the layouts, (encl) or order them from myself. The board and parts kits' prices are listed at the bottom of the parts list on page .

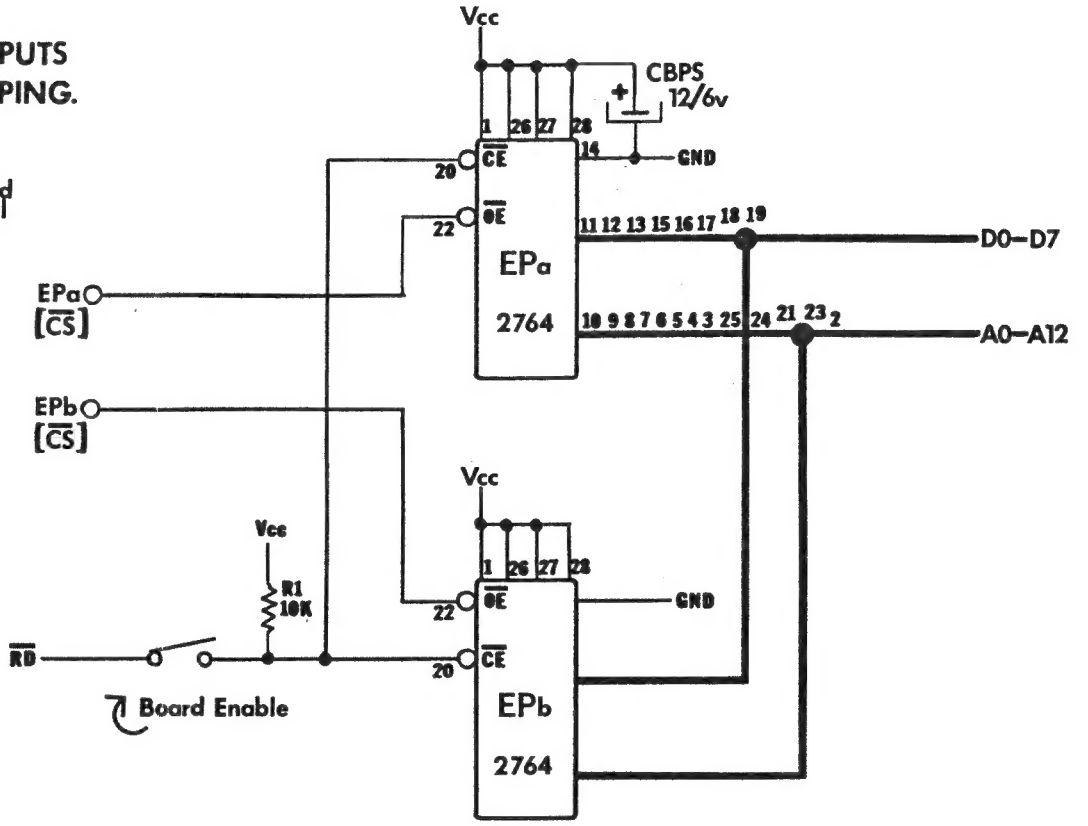
2764 x 2 EPROM READ

FOR INTEL PINOUT TYPE 2764s

Circuit will map 2-2764s ANYWHERE!

JUMPER EP_A + EP_B $\overline{CS}$ TO PROPER OUTPUTS OF THE 138 FOR DESIRED MEMORY MAPPING.





Connect pin 6, above, to M1 NOT if you are using the 48-64K block. If you are using the printer port (FFFFH) and this board mapped 48-64K, then instead of this jumper, connect with D1 + D2 as shown. If this board is mapped anywhere else, R2 will enable the 138 with no additional connections nec.

The 0-8K (0), 16-24K (16), and 24-32K (24) outputs of the 138 cannot readily be used on the ZX-81/TS1000 for obvious reasons. (Ram and Rom monitor)

R1 mounts right below the 138 and Cbps mounts right above pin 14 of EPb. SW1 mounts right below and to the right of R1, marked with an "x" between it's two mounting holes. D1 + D2 are mounted on the bottom of the board where convenient, if needed.

JLO 8308

2764 x 2 CARTRIDGE BOARD ASSY INSTRUCTIONS

STEP 1) Install feedthrough wires in all holes that have round "doughnut" pads on both the board top and bottom. The exception to this is where components mount, that use their leads as feedthroughs. These are: R1 on both it's ends, R2 on it's top end, and Cbps on it's top (+) end. You may cover these holes to remind yourself, if desired. Use no.30 solid wire or stripped wire wrap wire for the feedthroughs. There are not too many feedthroughs on this board, so this job should go fairly fast. After they are all in, remove all traces of flux with acetone and a soft cloth. Now inspect your work for shorts or bad joints and touch up if nec.

STEP 2) Install the 16 pin ic socket, with pin 1 down. Install the two 28 pin ic sockets with pin 1 on the right. (as viewed from the component side of the board with the edge connector traces down) After soldering in place, remove the flux with acetone and carefully inspect.

STEP 3) Now install the two resistors. Remember that R1's leads are soldered both on the top and bottom of the board, and R2's top lead is soldered also on top and bottom. Install one of the bypass capacitors on the board just right of R2, with it's red (pos) end toward the boards top, and soldered also both top and bottom. Install the other bypass cap under EPa, by forming it's leads to fit between pads 14 + 28. Be sure that it's red (pos) end is connected to pin 28.

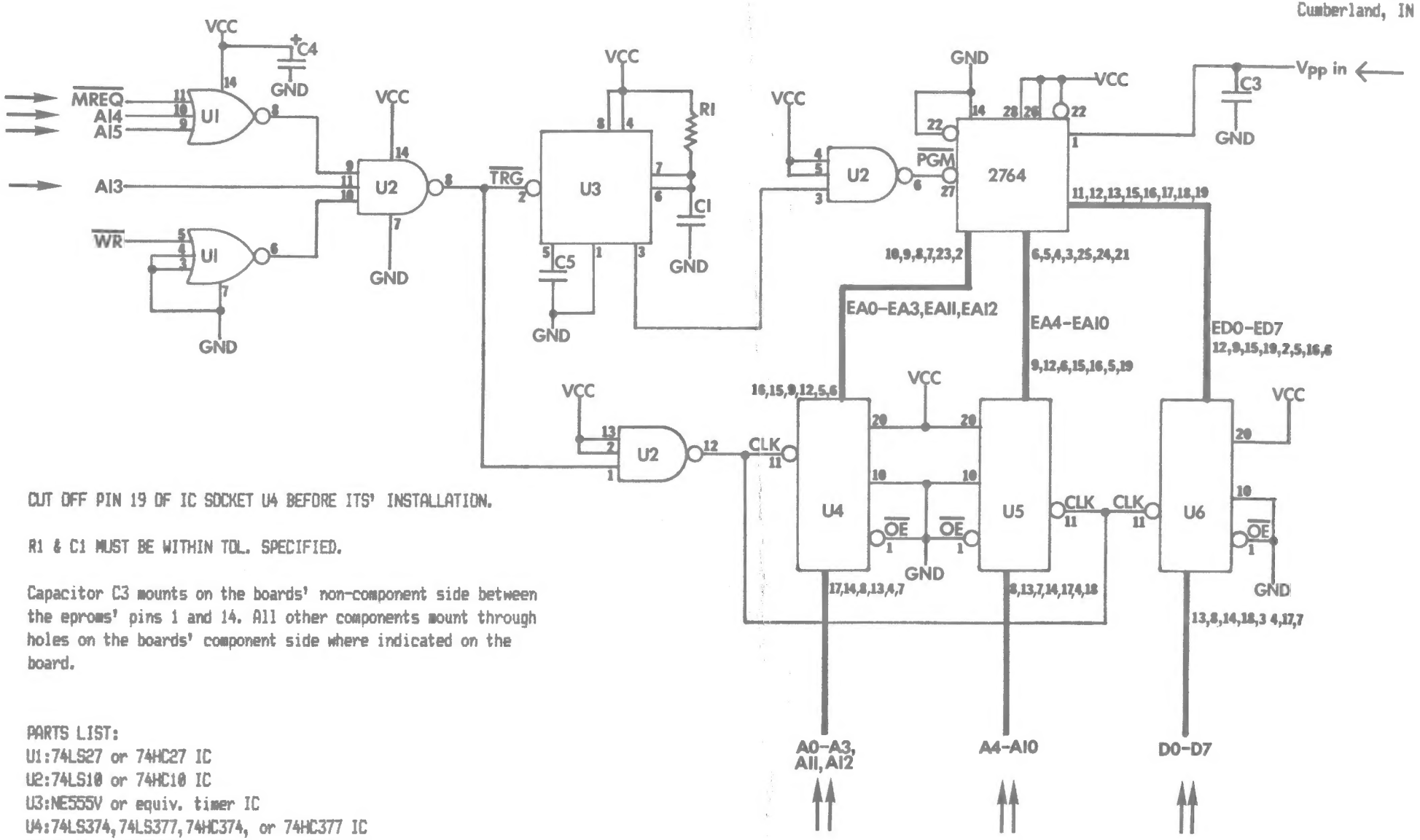
STEP 4) If you are using my printer port from SQ no.1, install diodes D1 + D2 on the boards bottom. Jumper diode D2 from its' cathode (band) end to edge connector trace 51B. The printer port's PORT SEL NOT output should also be tied to trace 50B, on it's board. Solder the wire to the very top of this edge trace. If you do NOT have my printer port, instead of the diodes simply jumper pin 6 of the LS138 to the pad marked M1. Remove flux from all these connections with a Q TIP and acetone, and carefully inspect your work.

STEP 5) Jumper pad EPa to the dedoder chip's "48" output (pin 9) and pad EPb to the decoder's "56" output (pin 7). Install SW1 in the holes marked on the board with an "x", then plug in the 74LS138 chip itself. The board is now ready to accept two 2764 eproms, and they will be mapped at the top of memory.

2764 EPROM PROGRAMMER FOR INTEL PINOUT 2764S. MAPPED 2000-3FFFh

FOR USE WITH THE ZX-81, TS1000, TS1500, OR TS2068 PERSONAL COMPUTER

(C) The John Oliger Co.
11601 Whidbey Dr.
Cumberland, IN 46229



CUT OFF PIN 19 OF IC SOCKET U4 BEFORE ITS' INSTALLATION.

R1 & C1 MUST BE WITHIN TOL. SPECIFIED.

Capacitor C3 mounts on the boards' non-component side between the eproms' pins 1 and 14. All other components mount through holes on the boards' component side where indicated on the board.

PARTS LIST:
U1:74LS27 or 74HC27 IC
U2:74LS10 or 74HC10 IC
U3:NE555V or equiv. timer IC
U4:74LS374, 74LS377, 74HC374, or 74HC377 IC
U5:74LS374, 74LS377, 74HC374, or 74HC377 IC
U6:74LS374, 74LS377, 74HC374, or 74HC377 IC
C1:.1mf 5% mylar capacitor
C2:Not used
C3:.1mf/25V or greater mono or disc cap
C4:12mf/6V or greater tantalum cap
C5:.1mf/10V or greater mono or disc cap
R1:453K/1% metal film resistor

JUMPER: U4-pin 5 to EP pin 23
U4-pin 6 to EP pin 2
U5-pin 1 to U4-pin 1

2764 PROGRAMMER ASSY INSTRUCTIONS

STEP 1) Install all feedthroughs using no.30 solid bare wire. A feedhrough should be installed in every hole that has a round "doughnut" pad on both the boards' top and bottom. If either pad is not round, do not install a feedthrough there because a component will mount there later. I have purposely layed this board out in this way as an aid in assy.

STEP 2) Install all the ic sockets, being sure to orient them in the correct direction. U1,U2,U3 and U6 install with pin 1 down while U4 and U5 mount with pin 1 to the left. The 28 pin eprom socket installs with pin 1 to the right. Be sure to cut off pin 19 on U4's socket before installing it. This pin is not used by this circuit. Remove flux with acetone and inspect.

STEP 3) Install C5, C1, R1, and C4 just left of U3, as indicated on the board. Be sure and solder the leads on both the top and bottom of the board, if there is a pad on both sides. Next install capacitor C3 on the boards' bottom, between pins 1 and 14 on the eprom socket. Use a pair of small round nose pliers to form its' leads to fit between the pads, cut off any excess lead length, and solder in place.

STEP 4) Prepare two 4" pcs. of no 22 solid insulated wire (1=red + 1=black) by stripping about 1/8" of insulation from one end of the wires and about 3/4" of insulation from the other end of the wires. Form the latter stripped end of each wire into an "O", then twist the two wires together. Now solder the shorter stripped ends as follows: Red wire to eprom socket pad 1 and black wire to eprom pad 14. Remove all flux and inspect.

STEP 5) Solder a jumper wire from U4-pin 1 to U5-pin 1, another from U4-pin 5 to eprom socket pin 23, and another jumper from U4-pin 6 to eprom socket pin 2. Remove flux with Q TIPS and acetone and inspect.

STEP 6) Install all of the ics in their correct sockets, making sure that both pin 1 is on the correct end and that all pins actually enter the socket contacts and do not curl up or stick out over the side of the socket. You may have to bend the pins inward slightly in order to insert them correctly. This completes this boards' assembly.

How to use the 2764 Programmer

The procedure for using the 2764 programmer is as follows:

Install the eprom to be programmed in the 28 pin socket, with pin 1 to the right. Plug the programmer into the expansion board and attach the Vpp wires from the power supply to the boards' Vpp input wires. (Red alligator clip to the red wire (Pos), and the black alligator clip to the black (Gnd) wire.

Verify that the Vpp supplys' 4.4/Vpp switch is in its' 4.4 position, and that the 21/25V switch is in its' 21V position (for a 2764) Now, power up the computer and after the curser appears, plug in the Vpp supply. The supplys' green "PWR" LED should light, but its' red "Vpp" LED should NOT light, at this point.

If we wish to put a listable program on eprom, then LOAD in the program from tape. If we wish to put data or a MC program on this eprom, then load your MC loader tape or start poking it in above ramtop. After all data is somewhere in RAM, then enter a BASIC program such as the typical BASIC eprom programming routine. This example routine will program the eprom with the contents of RAM, from C000-DFFFH (49152 to 57343 dec).

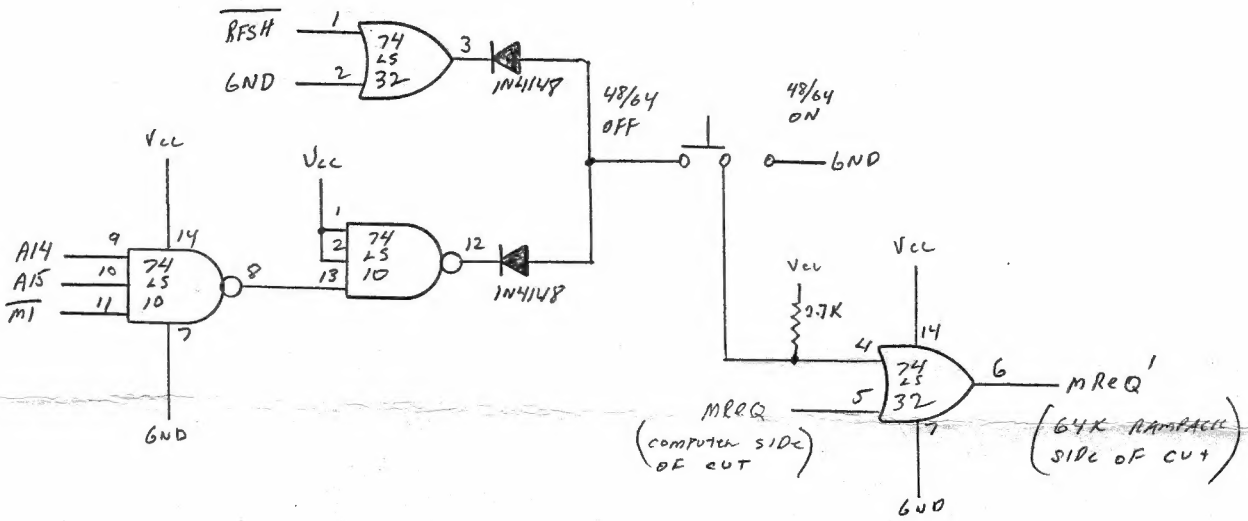
```
10 SLOW
20 REM X=Base address of programmer
30 LET X=8192
40 FOR N=49152 TO 57343
50 POKE X,PEEK N
60 LET X=X+1
70 PAUSE 4
80 NEXT N
```

After this program is keyed in, or a BASIC program is LOADed from tape if the MC programming routine is to be used, slide the 4.4/Vpp switch to the "Vpp" position on the Vpp power supply board. The red LED should now light. If you are using the BASIC routine, key in: GOTO 1/ENTER. If you are using the MC version, key in: RAND USR XXXX--where XXXX=the memory location of the routines' beginning. (8623 if mapped beginning at 21AFH, as mine is, in the listing) Then press ENTER. As soon as the programming is done, as known by the Timex/Sinclair report code appearing on the screen, slide the 4.4/Vpp switch back to its' 4.4 position. Now unplug the Vpp supply and then turn off the computer. Your program or data/MC routine is now stored on the eprom.

EXPANSION BOARD ERRATA

The 100 pin double read out edge connectors, as listed in SO#1 are no longer available. Because the expansion board in that issue was designed to take advantage of the availability of these very low priced connectors, and because of the fact you can no longer get them at a reasonable price, I have discontinued offering this board. In its' place, I offer the "4-SLOT" expansion board, shown in layout no. . This boards' physical size is more in line with the size of the ZX/TS, and is, in my opinion a better design. It is available from the author at the following prices, if you do not desire to etch your own board. BARE 4-SLOT EXPANSION BOARD.. $10.00pp USA or Canada only. Also available: 6 SLOT EXPANSION BOARD Holds six 50 pin soldertail edge connector sockets and six 20 pin ic "experimental" sockets. Bare board: $14.95ea pp--USA or Canada. Canadian orders must be paid by money order payeable in US dollars.

64 K RAM MODIFICATION TO TURN OFF THE 48-64K BLOCK OF RAM.
(NOT NEC. WITH THE 64K RAM IN SQ NO. 3)



CUT $\overline{MREQ}$ TRACE ON EXPANSION BOARD RIGHT BEFORE RAMPACK, & INSTALL CIRCUIT ABOVE.

EPROM CARTRIDGE DOWNLOAD ROUTINE

This routine will take a basic program stored in the 16K memory block from C000-FFFFh and move it down into the normal Basic ram area. (4009h and on) More than one program may be stored on a board in this block. POKE 16417 the number of the program desired to be downloaded.

The first programs' number is zero, the seconds' is one, etc. so that the routine will default on power up to download the first program in the block. (Because 16417=00 on power uo) The programs should be stored on eprom with no separating bytes. If the desired program, other than program zero, does not exist on the cartridge, the computer will return with report "E", program requested is not on the cartridge board. If there are no programs available, because the cartridge is not inserted into the expansion board, the cartridge board enable switch is off, or the memory board's 48/64K switch is on, the computer will crash.

```
2172 CDE702    STRT CALL TFAS      ;Enter "TEMP FAST" mode
2175 2100C0         LD HL,C000     ;Point 'HL' at cartridge slot
2178 110B00    AGIN LD DE,000B     ;Set 'DE' for displcmnt to ELINE
217B 19             ADD HL,DE      ;Point 'HL' at ELINE
217C 5E        EOPA LD E,(HL)      ;  'DE'=ELINE
217D 23             INC HL         ;  address
217E 56             LD D,(HL)      ;
217F 3A2140         LD A,(4021)    ;Get program no. to download
2182 A7             AND A          ;'A'=00?
2183 EB             EX DE,HL       ;'HL'=add ELINE/'DE'=ABS ELINE
2184 2019           JR NZ NEXT     ;Jump if wrong program
2186 010940         LD BC,VERSN    ;Set 'BC' for calculation
2189 C5             PUSH BC        ;Save 4009h on the stack
218A ED42           SBC HL,BC      ;'HL'=byte count
218C 44             LD B,H         ;  'BC'=byte
218D 4D             LD C,L         ;  count
218E EB             EX DE,HL       ;'HL'=ABS address ELINE+1
218F 110C00         LD DE,000C     ;Set 'DE' for calculation
2192 ED52           SBC HL,DE      ;'HL'=start of the program
2194 D1        DEST POP DE         ;'DE'=dest (4009h)
2195 EDB0           LDIR           ;Move the program
2197 E1             POP HL         ;Discard previous return address
2198 217606         LD HL,0676     ;Set HL for new RETurn address
219B E5             PUSH HL        ;Put new return address on stack
219C C30702         JP SLO?        ;Return to 0676h via SLO?
219F 011540    NEXT LD BC,4015     ;Set BC for calculation
21A2 ED42           SBC HL,BC      ;HL=no. of bytes to skip
21A4 19             ADD HL,DE      ;HL=start of next program
21A5 FD3521         DEC (IY+21)    ;Decrement program counter
21A8 7E             LD A,(HL)      ;Get first byte of next program
21A9 FEFF           CP FF          ;Is another program there?
21AB 20CB           JR NZ AGIN     ;Jump if there is
21AD CF             RST 08H        ;  Return w/report "E"
21AE 0D             REPORT E       ;  Program no. not available
```

BASIC PROGRAM PROGRAMMER ROUTINE

This routine will program a 2764 eprom with the current program in memory. The amount of data "SAVED" to eprom will be the same amount as if the normal SAVE command was used, so if you don't need the variables in a program, CLEAR before calling to save eprom space. The USR routine may be called from within a program, if desired, to make the program self running.

POKE 16507 and 16508 either: The location in the eprom you wish the routine to start programming at. (For more than one program on an eprom--default starting address is 2000h) or The location in ram you wish the routine to start from. (In case the program is greater than 8K in length and this is the second eprom to be programmed.) If the last eprom was entirely loaded with the beginning of the current program in memory, then POKE 16507,9 and POKE 16508,96--making the first address to be programmed from this time 6009h. Default source address used by this routine is 4009h. (VERSN) POKE 16507 with the lo byte of the 16 bit address and 16508 with the hi byte. (The same method used when changing ramtop, lo byte first + hi byte last) Be sure both 16507 + 16508 are zero if you are using the default addresses!!!

```
21AF CDE702    STRT CALL 02E7      ;Enter "TEMP FAST" mode
21B2 2A7B40         LD HL,(407B)   ;Get user pass bytes
21B5 AF             XOR A          ;Let 'A'=00
21B6 327B40         LD (407B),A    ;  Clear user
21B9 327C40         LD (407C),A    ;  pass bytes
21BC 7C             LD A,H         ;'A'=hi pass byte
21BD FE20      DFT? CP 20          ;Passed address )1FFF?
21BF 3005           JR NC SRC?     ;Jump for further tests if true
21C1 210940         LD HL,VERSN    ;'HL'=default src
21C4 180A           JR DEST        ;Continue at 21D0
21C6 FE40      SRC? CP 40          ;Address passed )3FFF?
21C8 3006           JR NC DEST     ;Jump if true. Addr. is for src
21CA EB             EX DE,HL       ;'DE'=passed dest
21CB 210940         LD HL,VERSN    ;'HL'=default src
21CE 1803           JR CONT        ;Continue at 21D3
21D0 110020    DEST LD DE,2000     ;'DE'=default dest
21D3 EB        CONT EX DE,HL       ;'DE'=src    'HL'=dest
21D4 E5             PUSH HL        ;Save dest
21D5 2A1440         LD HL,(ELINE)  ;Get last byte to move-1
21D8 A7             AND A          ;Clear carry
21D9 ED52           SBC HL,DE      ;'HL'=byte count
21DB 44             LD B,H         ;  'BC'=byte
21DC 4D             LD C,L         ;  count
21DD EB             EX DE,HL       ;'HL'=src
21DE D1             POP DE         ;Restore dest
21DF EDA0      MBYT LDI            ;Program a byte
21E1 E20702         JP PO SLO?     ;Return via "SLO?" if done
21E4 E5             PUSH HL        ;Save src
21E5 2100C0         LD HL,C000     ;Set 'HL' for test
21E8 19             ADD HL,DE      ;Dest)3FFF?
21E9 3002           JR NC NOER     ;Continue at 21ED if not
21EB CF             RST 08H        ;  Return with report "G"
21EC 0F              REPORT G      ;  Dest)3FFF
21ED D5        NOER PUSH DE        ;Save dest
21EE C5             PUSH BC        ;Save byte count
21EF 210400         LD HL,0004     ;Set 'HL' for the pause
21F2 CD2D02         CALL 022D      ;PAUSE 4
21F5 FD3635FF       LD (IY+35),FF  ;Set FRAMS-hi
21F9 CD4B0F         CALL 0F4B      ;Set DBNC
21FC C1             POP BC         ;Restore byte count
21FD D1             POP DE         ;Restore dest
21FE E1             POP HL         ;Restore src
21FF 18DE           JR MBYT        ;Loop to send another byte
```

2764 EPROM PROGRAMMER Theory of Operation

Two 3-input NOR gates and one 3-input NAND gate form an address decoder for the addresses from 2000H-3FFFH. If A15, A14, MREQ NOT, and WR NOT are all low into the two NOR gates, then the output of both of these gates will be high. If A13 is also high, along with the these two outputs, then the output of U2 at pin 8 will go low. Consequently, if the CPU is addressing any byte from addresses 2000H-3FFFH, and is wanting to WRITE to memory (as opposed to writing to an I/O device), then U2-pin 8 goes low. This active low signal is applied to U2-pin 1 and U3-pin 2. At U2-pin 1 the signal is inverted because this gate is wired as an inverter, and emerges at pin 12 as an active high signal. This signal is applied to the three octal flip flops at their CLOCK pin, which forces the flip flops to transfer the then valid addresses and data from thier inputs to thier outputs. The flip flops will hold these addresses and data until they recieve another low to high transition on thier CLOCK inputs. (IE. Until another POKE to this block of memory.)

Meanwhile, the active low signal from U2-pin 8 has also been applied to U3-pin 2. Pin 2 of U3 is this timers' TRIGGER NOT input, so after a typical delay through the chip of 100ns, (fulfilling the eproms' stable address/data line preprogram requirement) U3s' output at pin 3 goes high and stayes high for aprox. 50ms, because of the values of R1 and C1. This active high timed pulse is inverted by U2-pin 3--U2-pin 6 to become an active low timed pulse. This signal is then applied to the eproms' PGM NOT input, pin 27, which triggers the eprom to program the location pointed to by its' address lines the contents of its' data lines if: 1) Vpp (21VDC) is applied to pin 1 and 2) All the address/data lines remain stable for at least 50ms.

The latter requirement is of course easily met by the octal flip flops, as long as there is not another POKE to this block for at least 50ms, hence the PAUSE 4 command in the Basic programming routine. Because the stable address and data line requirement is met via the use of these flip flops, rather than by using the Z80 WAIT NOT input line, any dynamic memory connected to the system will NOT be corrupted by inadequate refreshes caused by excessive WAITing. Thus this programmer is completly compatible with any rampack you may be using.

The only requirement the circuit needs, as far as software is concerned is to be sure that the memory block this eprom is mapped at is not written to again, after recieving data, for at least 50ms. This is to allow time for the eprom to complete its' cycle. This is very easily accomplished from Basic with the PAUSE command. A PAUSE 4, after each POKE to this block, will delay the system 66ms, more than enough time. The programmer is just a programmer, not an eprom reader. Thus, if you POKE one of it's memory locations, it is permanently stored in the eprom, if Vpp is applied. If Vpp is NOT applied, then no programming can take place, but no harm will come to the eprom either. Because the board is decoded with WR NOT, you could have eprom mapped at the very same location, decoded with the RD NOT signal, and transfer the data from eprom to eprom with a Basic line like POKE N, PEEK N.

2764 READ BOARD Theory of operation

If MREQ NOT is active (low) and PORT SEL NOT (if used) is high (will be pulled high by R2 if not used), the 74LS138 Binary decoder chip will sample it's input lines, connected to addresses 13, 14, and 15, and depending on the binary value of these address lines, bring one of its' output terminals active (low). Thus, the decoder chip is capable of selecting just where in memory a particular eprom is mapped at, via the use of a jumper to its' appropriate output pin.

If the boards' ENABLE switch is switched on, the CPUs' RD NOT signal is applied to the eproms' CS NOT input. Then, if RD NOT goes active, and after this the OE NOT input goes active from the decoder chip, the eprom takes the contents of the address pointed to with address lines A0-A12, and puts it on the data bus. The access requirement of the circuit is determined from the time that RD NET goes active low, rather than when OE NOT goes low, because of the way the eprom is constructed. (rated access time is typically 2 times faster from OE NOT to valid data verses CS NOT to valid data).

HOW TO PROGRAM A 2764 EPROM USING "HOT Z" (C), TO STORE A ZX-81 PROGRAM IN HIGH MEM (COOOH) FOR MC DOWNLOADING.

The program to be stored on eprom must have been previously saved to tape. The eprom will be programmed exactly as it is stored on tape, so if you don't need variables, clear before SAVING to tape to save eprom space.

1) Load or run "HOT Z".
2) Enter C009 in read mode, hit "Y", hit "P", enter FFFF.
3) Hit "J", start tape for Loading, then hit "ENTER".
4) When pattern on CRT screen shows the entire program has been loaded, hit "BREAK". (space)
5) Look at locations C014 + C015.
6) Enter the 2 digets shown in C015, then the 2 digets shown in C014-1. (EG. If C014=8F, and C015=5D, enter 5D8E.)
7) You are now at a location in the 4000-7FFF block. Add 8000h to the number you just entered and enter it. (EG. In our example, you would enter DD8E, because 5D8E+8000=DD8E.)
8) You are now at at location in the C000-FFFF block. Hit "K" to find the decimal equivalent of the address we are at now, and write it down. (EG. In our example, we would write down 56718.)
9) Now hit "Q" to quit to Basic. The 2764 programmer should have already been set up + ready to go, with an eprom in it.
10) Enter a typical Basic eprom programming program. Use 49161 for the first number in the for/next loop, and the number you wrote down for the second number. (EG. In our example, you would have FOR N=49161 TO 56718 for this line.) If the number you previously wrote down is greater than 57352, then you will have to program two eproms to hold the program, as it is longer than 8K. Use 57352 as the second number this time, and use 57353 for the first number of the FOR/NEXT loop next time. You will have to load in the program again at C009 with "HOT Z", and program another eprom with the rest.
11) Turn on Vpp to the programmer, and enter GOTO 1. Your eprom will now be programmed.
12) If there is already another program on the eprom, simply enter LET X=(next avail. bytes' location) for the line setting the programmers' initial starting address. Do not seporate the programs with anything. The downloader program will calculate where the program begins on downloading. Be sure, if you do already have a program on the eprom, that you have enough room left on it to hold the new program. A line like IF X)=16384 THEN STOP in the programming program will stop the routine from going too far.